RECON: RETRIEVAL-CONDITIONED NORM CALIBRATION FOR LIFELONG MODEL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Lifelong model editing requires a language model to absorb a growing sequence of factual updates while preserving prior knowledge and unrelated behavior. Residual-memory editing enables such scalable updates by storing each edit as a sparse residual. At scale, however, the hash-based readout of existing residual-memory editors induces two failure modes: routing fragility, because hash lookup is not invariant under paraphrase, and norm inflation, because collisions on shared columns become inevitable over long edit sequences. We present RECON, which repairs the readout with two complementary training-free mechanisms. First, semantic routing–support decoupling replaces rigid hash lookup with a gated dense semantic key while executing the sparse support verbatim, restoring robust retrieval across paraphrased queries. Second, retrieval-conditioned norm calibration rescales each retrieved dimension by a power of its relative column norm, directly counteracting collision-induced magnitude distortion to sustain locality under sequential edits. Across temporal, hallucination-correction, and sequential editing benchmarks, RECON nearly doubles out-of-distribution generalization over the hash-based readout at $T{=}10$, preserves locality across all evaluated horizons, and retains its advantages as edit sequences grow.

1 INTRODUCTION

Large language models (LLMs) can be updated post hoc to incorporate new factual knowledge without full retraining. In *lifelong knowledge editing*, updates arrive sequentially, and the editor must continuously absorb new facts while preserving previously acquired edits and unedited behavior. As the editing horizon extends, cumulative interactions among updates cause severe interference, a process akin to catastrophic forgetting (Kirkpatrick et al., 2017) that makes long-horizon stability and generalization difficult to sustain.

Existing editors broadly follow two paradigms. In-place methods update model weights directly, through hypernetworks (Cao et al., 2021; Mitchell et al., 2022a), closed-form projections (Meng et al., 2022; 2023), or null-space constraints (Fang et al., 2025); memory-based editors store edits in auxiliary structures and route queries at inference (Mitchell et al., 2022b; Huang et al., 2023; Hartvigsen et al., 2023; Wang et al., 2024a). Residual-memory editing (Wang et al., 2025) bridges the two paradigms: it freezes the base model and stores each edit as a sparse residual in a shared parameter space, so each edit touches only a small subset of parameters, direct interference is substantially reduced, and thousands of edits accumulate in a single memory.

Despite this structural advantage, residual-memory editing still degrades substantially as the editing sequence grows. Because the base weights are frozen and every injected update is structurally preserved, this degradation cannot originate from overwriting during the *write* phase; the bottleneck instead shifts to the *readout*: how accumulated memories are retrieved and executed at inference. Examining the readout interface, we identify two failure modes that compound at scale.

First, memory retrieval suffers from **routing fragility** rooted in discrete write allocation. Existing residual-memory editors multiplex a single hash-based representation for both write allocation and inference readout. While hashing disperses edits effectively across the residual space, retrieval demands a fundamentally different property: *semantic invariance* across paraphrases and distribution shifts. A successfully written edit thus becomes unreachable once the query is reworded: the wording changes, and the hash changes with it (e.g., "The telephone was invented by Alexander Graham

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

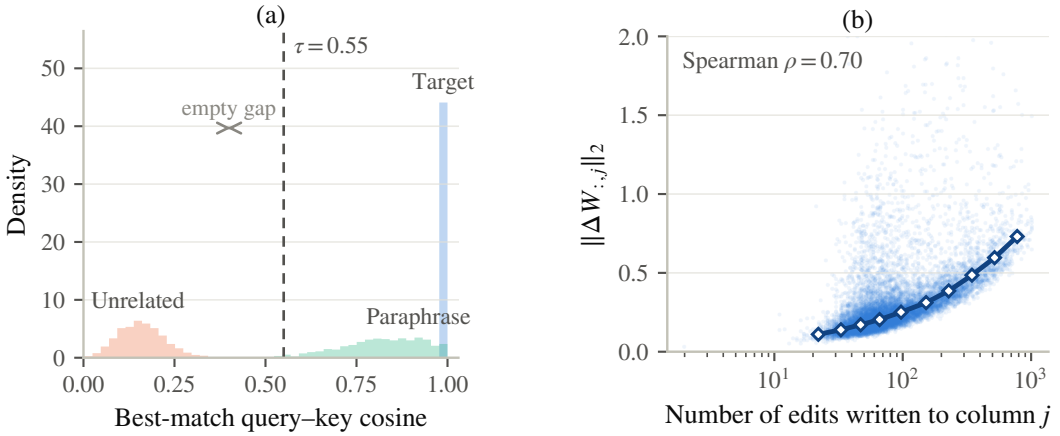

Figure 1: Readout diagnostics on ZsRE, a question-editing benchmark, with LLaMA-3-8B at $T = 1{,}000$. (a) Distributions of the best-match query–key cosine similarity for target, paraphrased, and unrelated queries: paraphrases remain close to their stored keys while unrelated queries stay strictly below, so any gate threshold inside the empty band separates the two regimes. (b) Residual column norm $\|\Delta W_{:,j}\|_2$ against the number of edits written to each column (Spearman $\rho = 0.70$): write collisions inflate the same columns that readout executes.

Bell" versus "Who created the first device to transmit speech?"). Crucially, the loss is incurred at readout rather than in storage (Figure 1a): executing the top-ranked memory without the acceptance gate keeps ZsRE paraphrase generalization at $T = 1$ at essentially its gated ceiling (≈ 0.99 versus 1.00). Stored residuals therefore transfer almost perfectly across reformulations; degradation at longer horizons arises in how the accumulated memory is read out, not in the quality of the individual edits learned.

Second, repeated reuse of residual dimensions causes **norm inflation** during execution. As sequential edits accumulate in the shared residual matrix, write collisions fall unevenly across columns, making their column norms $\|\Delta W_{:,j}\|_2$ increasingly heterogeneous. Direct readout then grants disproportionate leverage to over-written dimensions, allowing magnitudes inflated by unrelated past edits to corrupt current outputs. The distortion is systematic rather than incidental: at $T = 1{,}000$, a residual column's norm correlates strongly with the number of edits written to it (Spearman $\rho = 0.70$; Figure 1b), so the dimensions most corrupted by collisions are the ones executed most strongly. A scalable editor must therefore govern not only *which* memory is retrieved but also *how* the retrieved residual is executed.

These observations call for an explicit decoupling of memory retrieval from residual execution. We present **RECON** (**RE**trieval-**CO**nditioned **N**orm calibration), which repairs the readout with two complementary, training-free mechanisms:

- **Semantic routing–support decoupling** addresses routing fragility by replacing rigid hash lookup with a gated dense semantic key while executing the stored sparse support verbatim, so retrieval relies on continuous semantic similarity rather than brittle discrete hashing. The gate threshold is placed above the empty band of Figure 1a: unrelated queries stay strictly below it, and only a negligible fraction of paraphrases falls under the threshold.
- **Retrieval-conditioned norm calibration** counteracts norm inflation by rescaling each retrieved dimension by a power of its relative column norm in closed form, directly targeting the collision-driven magnitude heterogeneity of Figure 1b without modifying stored parameters.

RECON leaves the underlying sparse write mechanism untouched: each edit is still learned as a residual over a hash-allocated support, and what changes is how the stored residuals are accessed. RECON redesigns the readout rather than the way new knowledge is learned, mitigating the degradation that emerges during long-horizon memory reuse while retaining the advantages of sparse residual editing.

We evaluate RECON on out-of-distribution temporal editing, where the retrieval gap is largest; on sequential editing on ZsRE, up to single sequences of $7{,}000$ and $15{,}000$ edits; and on hallucination correction, which goes beyond factual question answering. RECON consistently improves over the hash-based readout, nearly doubling paraphrase generalization under temporal shift (0.72 versus 0.37 at $T{=}10$) while sustaining locality of 1.00 through $15{,}000$ sequential edits; diagnostic analyses tie the gains to the two failure modes they target.

Our contributions are summarized as follows:

- **Semantic routing–support decoupling.** We separate the representation used for memory retrieval from the sparse support used for residual execution: a training-free semantic key provides robust routing across paraphrases and distribution shifts, while the original sparse support preserves the structure produced by the editing process.
- **Retrieval-conditioned norm calibration.** We introduce a lightweight, closed-form calibration mechanism that adjusts the contribution of retrieved residual dimensions according to their accumulated magnitudes, suppressing collision-inflated dimensions during memory execution without modifying stored parameters.
- **Scalable lifelong editing.** We demonstrate that readout repair substantially improves the robustness of residual-memory editing over long sequences and under distribution shift, with locality preserved at 1.00 through $15{,}000$ sequential edits.

2 Related Work

Knowledge editing changes the factual behavior of a pretrained model without full retraining (Yao et al., 2023; Zhang et al., 2024; Wang et al., 2024b). In-place methods update weights directly: hypernetwork editors (Cao et al., 2021; Mitchell et al., 2022a) map editing requests to parameter updates, locate-then-edit methods (Meng et al., 2022; 2023) solve closed-form updates to selected MLP projections, and AlphaEdit (Fang et al., 2025) constrains updates to the null space of preserved knowledge; sequential application overwrites shared parameters and compounds interference, the failure mode behind catastrophic forgetting (Kirkpatrick et al., 2017). Memory-based editors instead store edits outside the original parameters and route queries at inference: SERAC (Mitchell et al., 2022b) adds an external memory with an auxiliary head, T-Patcher (Huang et al., 2023) attaches a neuron per correction, GRACE (Hartvigsen et al., 2023) overwrites activations on discrete-key matches, WISE (Wang et al., 2024a) trains a routed side memory, MELO (Yu et al., 2024) and RECIPE (Chen et al., 2024) index low-rank adapters and knowledge prompts, LEMoE (Wang & Li, 2024) routes edits among mixture-of-experts adaptors, and RLEdit (Li et al., 2025) retrains the editor with reinforcement learning. These designs insulate the base model from sequential interference, but they introduce trained components and, in most designs, per-edit storage growth, and retrieval depends on learned classifiers or discrete keys whose stability under paraphrase and shift is hard to guarantee.

Residual-memory editing keeps the base model frozen and stores each edit as a sparse residual over a target projection, the storage pattern underlying adapter-based adaptation (Hu et al., 2022); sparse allocation limits overlap, so one memory absorbs thousands of edits. MEMOIR (Wang et al., 2025) is the strongest editor of this kind published so far, but it couples writing and reading through a single hash-based representation: the hash that allocates supports also retrieves memories, so stored knowledge becomes inaccessible under paraphrase or shift, and heavily written columns develop inflated norms that distort execution. RECON addresses both failures in the readout while leaving the residual training untouched: semantic routing decouples retrieval from sparse allocation through a training-free key, and norm calibration corrects the accumulated magnitude profile in closed form, in the spirit of magnitude-decomposed adaptation (Liu et al., 2024) but applied post hoc.

3 Method

3.1 Problem Formulation

We consider lifelong knowledge editing, where a language model is sequentially updated with a growing stream of factual editing requests $\mathcal{E} = \{(e_i, y_i)\}_{i=1}^{T}$, with e_i the i-th request and y_i its

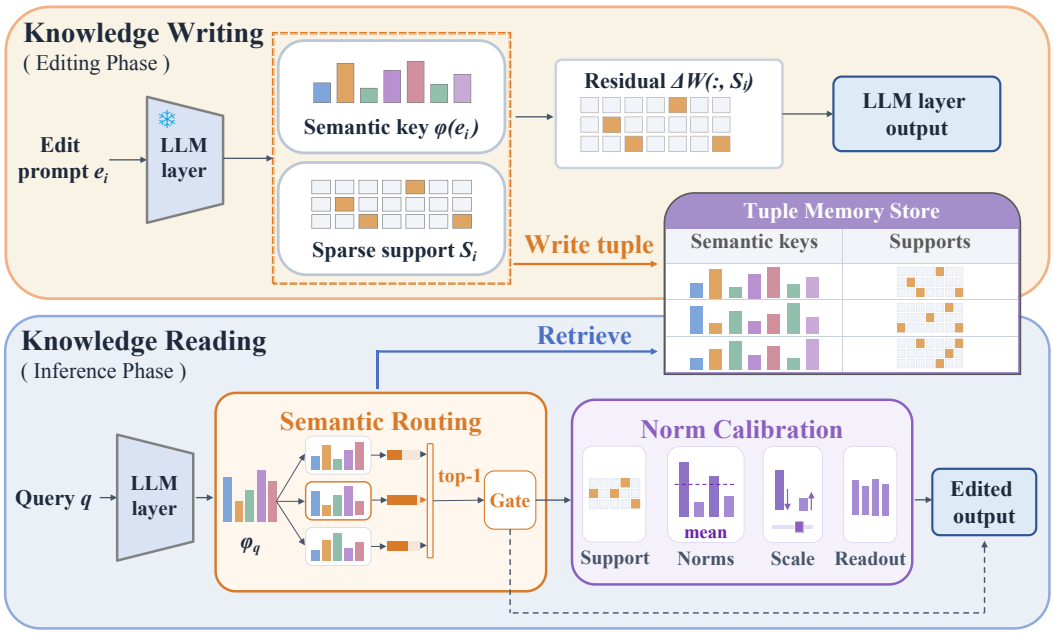

Figure 2: Overview of RECON. In the editing phase (top), each edit is learned as a residual $\Delta W_{:,S_i}$ over a hash-allocated sparse support S_i, and the tuple $(\varphi(e_i),\ S_i,\ \Delta W_{:,S_i})$ is written directly into the tuple memory store, whose columns hold the semantic keys and the supports (the residual entry is a view of the single shared matrix ΔW). In the inference phase (bottom), semantic routing scores the query representation $\varphi(q)$ against all stored keys and selects the top-ranked memory, a gate decides whether the residual branch is activated (with fallback to the base model otherwise), the stored support determines which residual dimensions are executed, and norm calibration rescales their contributions by the accumulated column norms. RECON redesigns only this readout; the way new knowledge is learned is unchanged.

desired completion. After T edits, the model should recall the edited facts (*reliability*), generalize to semantically equivalent queries (*paraphrase generalization*), and preserve its behavior on unrelated inputs (*locality*).

We consider an editor that learns new knowledge via hash-allocated sparse residuals. Let $W_0 \in \mathbb{R}^{d\times D}$ denote the target projection of the base language model and let $\Delta W \in \mathbb{R}^{d\times D}$ denote a residual matrix. The base model is kept fixed; edits accumulate into the shared matrix ΔW, $x(t) \in \mathbb{R}^D$ is the layer input at token position t, and $y(t) \in \mathbb{R}^d$ the projection output. Only the slice of ΔW indexed by a retrieved support is ever added to the base projection at inference.

For each edit i, a hash $h(e_i)$ of the edit prompt allocates a sparse support $S_i \subseteq \{1,\dots,D\}$ with $|S_i| = \kappa$, and the new fact is learned as a residual update ΔW_i over that support, accumulated in place following the standard residual editor (Wang et al., 2025) (objective and optimizer settings in Appendix A):

$$\Delta W \leftarrow \Delta W + \Delta W_i, \qquad \mathrm{supp}(\Delta W_i) = S_i. \tag{1}$$

All edits share the same matrix, and different edits may therefore overlap on a subset of columns.

At inference the baseline retrieves with the same hash representation: it scores each stored support by hash overlap with the query, executes the best match, and falls back to the base model otherwise,

$$y(t) = W_0\, x(t) + \Delta W_{:,S^{\mathrm{hash}}(q)}\, x_{S^{\mathrm{hash}}(q)}(t), \tag{2}$$

where $x_S(t)$ collects the activations of $x(t)$ on the columns of S and $\Delta W_{:,S}$ denotes the corresponding column slice: allocation and retrieval are two uses of one representation, the coupling that RECON removes. This coupling induces the two readout pathologies of the introduction. **Routing fragility:** the hash is not invariant to rewording, so for a paraphrased query the hash overlap

with the stored support can drop below the gate, leaving a successfully written residual unreachable. **Norm inflation:** write collisions fall unevenly across shared columns, and columns that absorb more writes develop larger norms, so direct execution grants disproportionate leverage to the most collided dimensions.

This formulation separates editing from readout: allocation and optimization determine *what* is stored, whereas inference must determine *which memory to retrieve*, *which dimensions to execute*, and *how strongly*. RECON addresses these three decisions explicitly, without altering how the residuals are learned.

3.2 Framework Overview

As illustrated in Figure 2, RECON keeps the write phase unchanged and redesigns only the readout, splitting the pipeline along the two panels.

Write phase (top panel). Each edit i is allocated a hash-based sparse support S_i and learned as a residual $\Delta W_{:,S_i}$ over that support; in addition, RECON computes a dense semantic key $k_i = \varphi(e_i)$. Each edit is thus stored as a tuple

$$M_i = (k_i,\ S_i,\ \Delta W_{:,S_i}) \tag{3}$$

in the tuple memory store, pairing the learned residual with its semantic key and sparse support. Here $\Delta W_{:,S_i}$ is the slice of the single shared residual matrix $\Delta W \in \mathbb{R}^{d\times D}$ of Section 3.1 restricted to S_i, a view rather than an independent copy. Edits accumulate in place in ΔW; the per-edit overhead of the memory store is only the metadata (k_i, S_i). Collisions are therefore physical: two stored memories interfere exactly when their supports overlap within ΔW. Parameter storage is consequently independent of the editing horizon ($O(d \times D)$), and the auxiliary metadata grows as $O(T\,(D + \kappa))$.

Readout phase (bottom panel). Given a query q, the readout proceeds in two stages. First, *semantic routing*: the query representation $\varphi(q)$ is compared against all stored keys, the top-ranked memory $i^*(q)$ is selected, and a gate decides whether the residual branch is activated at all, with fallback to the base model otherwise. Second, *calibrated execution*: the stored support $S(q) = S_{i^*(q)}$ is read out verbatim, and each of its dimensions is rescaled by a function of its accumulated column norm before entering the forward pass. The key and the support thus play distinct roles: k_i encodes the semantic identity of the edit and determines *which memory* is retrieved; S_i records the optimized dimensions and determines *which* are executed, and the calibration determines *how strongly*. This split targets the two pathologies of Section 3.1: semantic routing restores access when allocation representations are not invariant to paraphrase or shift, and norm calibration corrects the magnitude imbalance created by heterogeneous write histories.

3.3 Semantic Routing and Support-Preserving Execution

For each edit e_i, RECON constructs a dense semantic representation from the layer-input hidden states. The semantic key $k_i \in \mathbb{R}^D$ is obtained by averaging the hidden states over the token positions of the edit prompt and removing a background mean:

$$k_i = \varphi(e_i) = \frac{1}{|e_i|}\sum_{t\in e_i} x(t) - \mu, \tag{4}$$

where $|e_i|$ is the prompt length in tokens and μ is a background mean estimated from edit-irrelevant text. The key is stored with the edit and used exclusively for retrieval; it plays no role in constructing or optimizing the sparse support.

Given a query q, its semantic representation $\varphi(q)$ is computed using the same procedure. RECON retrieves the memory with the highest semantic similarity:

$$i^*(q) = \arg\max_i\ \cos(\varphi(q),\ k_i)\,. \tag{5}$$

To prevent unrelated queries from activating an edited memory, we introduce a semantic gate with threshold τ:

$$g(q) = \mathbb{I}\big[\cos\big(\varphi(q),\ k_{i^*(q)}\big) \geq \tau\big]\,. \tag{6}$$

If the similarity does not exceed τ, the residual branch is disabled and the model falls back to the base model. When the query is accepted, the stored support is executed verbatim, $S(q) = S_{i^*(q)}$: RECON never reconstructs a support from the query, which frees the routing representation from the requirements of sparse allocation and lets retrieval favor semantic invariance.

3.4 Retrieval-Conditioned Norm Calibration

Semantic routing determines which residual memory should be accessed, but does not account for magnitude differences accumulated in the shared residual matrix. Let

$$n_j = \|\Delta W_{:,j}\|_2 \tag{7}$$

denote the norm of residual column j. For a retrieved support $S(q)$, we compute its mean column norm as

$$\bar{n}(q) = \frac{1}{\kappa} \sum_{j \in S(q)} n_j. \tag{8}$$

RECON assigns each selected column a relative magnitude correction:

$$a_j(q) = \left(\frac{\bar{n}(q)}{n_j + \epsilon} \right)^{\alpha}, \quad j \in S(q), \tag{9}$$

where ϵ is a small numerical constant and $\alpha \geq 0$ controls the strength of the correction.

The calibrated residual contribution at token position t is then

$$\Delta y(t) = g(q)\, \Delta W_{:,S(q)} \left(a_{S(q)}(q) \odot x_{S(q)}(t) \right), \tag{10}$$

where $a_{S(q)}(q) \in \mathbb{R}^{\kappa}$ stacks the calibration factors $\{a_j(q)\}_{j \in S(q)}$ and $x_{S(q)}(t) \in \mathbb{R}^{\kappa}$ collects the activations of $x(t)$ on the retrieved support ($\kappa = |S(q)|$), so that $\odot$ denotes their elementwise product.

The exponent α trades off preserving original magnitudes against correcting accumulated imbalance: $\alpha = 0$ recovers the uncalibrated readout, and increasing α progressively suppresses relatively large columns, a controlled correction of the magnitude profile rather than full normalization.

The calibration is conditioned on the retrieved support rather than applied globally, adapting to the magnitude distribution of the accessed memory while leaving stored parameters unchanged.

3.5 Inference Procedure

Given a query q, the three components operate in sequence: semantic routing selects the memory $i^*(q)$ and its support $S(q) = S_{i^*(q)}$; the gate $g(q)$ decides whether the residual branch is activated at all; and norm calibration determines the strength of each executed dimension. The output of the edited projection is

$$y(t) = W_0\, x(t) + g(q)\, \Delta W_{:,S(q)} \left(a_{S(q)}(q) \odot x_{S(q)}(t) \right). \tag{11}$$

When $g(q) = 0$, the residual branch is inactive and the original computation $y(t) = W_0\, x(t)$ is recovered.

Routing is likewise anchored to the prompt: $\varphi(q)$ aggregates the prompt prefix only, so the retrieved memory, gate, and calibration factors are fixed by the prompt and constant across decoding steps; generated tokens never enter the key.

Readout overhead. The auxiliary computation is light. Semantic routing is a single top-1 cosine search over the T stored keys, that is, one $T \times D$ matrix–vector product per query, and the gate reuses its similarity score, so routing adds no additional model evaluation; norm calibration rescales only the κ retrieved dimensions in closed form. The exhaustive search adds under one millisecond per query up to T=1,000 and grows linearly to 12.8 ms at T=15,000, about 43% of one forward pass, against 2.24 ms for hash-overlap scoring at that scale (Appendix C); being a pure similarity search over static keys, it admits sub-linear approximate nearest-neighbor indexes. The key cache occupies $T \times D$ scalars, roughly 0.86 GB at T=15,000 for LLaMA-3-8B (about 5% of the backbone weights), and admits half-precision or quantized storage.

Table 1: Out-of-distribution editing on Temporal with GPT-J-6B. Each edit pairs a Wikipedia prefix with a passage about the same entity; reliability is measured on the edit prompts, generalization on the continuation of the same entries, and locality on unrelated passages from the Pile. The *w/o Editing* row reports the zero-shot base model.

	$T = 10$				$T = 75$			
Method	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
w/o Editing	0.56	0.21	–	–	0.56	0.21	–	–
MEMIT	0.88	0.28	0.98	0.71	0.41	0.18	0.14	0.24
GRACE	0.97	0.28	1.00	0.75	0.97	0.28	1.00	0.75
WISE	0.99	0.36	0.98	0.78	0.96	0.37	1.00	0.78
AlphaEdit	0.89	0.27	0.99	0.72	0.85	0.27	0.96	0.69
MEMOIR	0.99	0.37	1.00	0.79	0.99	0.38	1.00	0.79
RECON	0.96	0.72	1.00	**0.89**	0.96	0.49	1.00	**0.82**

4 Experiments

4.1 Experimental Setup

We evaluate RECON on three complementary benchmarks: OOD temporal editing on Temporal with GPT-J-6B (Wang & Komatsuzaki, 2021), sequential factual editing on ZsRE (Levy et al., 2017) with LLaMA-3-8B (Team, 2024) and Mistral-7B (Jiang et al., 2023), and hallucination correction on SelfCheckGPT (Manakul et al., 2023), with edit sequences from a single edit to $15{,}000$. We follow the grouped protocol of Wang et al. (2025): the pool is partitioned into groups of T sequential edits, the model is reset between groups, and all metrics are measured after each group's full edit sequence. Generalization on Temporal uses temporally shifted queries; on SelfCheckGPT, reliability is desired-completion perplexity (lower is better) and locality is token exact match. Baseline results are from Wang et al. (2025) under the same protocol, spanning in-place parameter editors (ROME, MEMIT, AlphaEdit), retrieval-gated memories (GRACE, WISE, DEFER), and sparse residual storage (MEMOIR); benchmark rationale, dataset and metric details, and implementation notes appear in Appendix A.

4.2 Out-of-Distribution Generalization

Table 1 evaluates OOD generalization on Temporal, which stresses the readout most: an edit that succeeds on the original prompt may still fail to be retrieved under temporal shift. RECON nearly doubles OOD generalization over MEMOIR at $T = 10$, raising it from 0.37 to 0.72, and retains a clear advantage at $T = 75$ (0.49 versus 0.38), with the best average at both scales (0.89 and 0.82 versus 0.79), reliability 0.96, and locality 1.00; the absolute margin narrows as T grows, but the advantage persists.

4.3 Sequential Editing

Standard sequential editing. Table 2 reports the standard sequential protocol on ZsRE, where queries match the editing distribution and retrieval is comparatively easy. RECON matches or exceeds MEMOIR at every scale on both backbones with locality at 1.00 (0.94 versus 0.93 at $T = 1{,}000$ on LLaMA-3-8B; 0.95 versus 0.94 on Mistral-7B). The margins are smaller than under distribution shift: lexical overlap keeps hash-based retrieval competitive in distribution.

Extreme-scale sequential editing. Table 3 extends ZsRE to single continuous sequences of $7{,}000$ and $15{,}000$ edits on LLaMA-3-8B, following Wang et al. (2025). RECON's average declines only from 0.93 to 0.92 while MEMOIR drops from 0.92 to 0.88 and parameter-modifying methods fail almost entirely; locality stays at 1.00. The one trailing cell, generalization at $T = 7{,}000$ (0.84 versus 0.85, within a point), recovers at $T = 15{,}000$ (0.82 versus 0.78).

Table 2: Sequential editing on ZsRE with LLaMA-3-8B (top) and Mistral-7B (bottom), under the grouped protocol that resets the model between groups of T edits. Reliability is measured on the edit prompts, generalization on their paraphrases, and locality on unrelated questions, with the average taken over the three metrics.

	$T = 1$				$T = 10$				$T = 100$				$T = 1{,}000$			
Method	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
LLaMA-3-8B																
FT	0.49	0.49	0.65	0.54	0.18	0.15	0.12	0.15	0.13	0.11	0.02	0.09	0.13	0.12	0.01	0.09
ROME	0.99	0.96	0.96	0.97	0.61	0.59	0.41	0.54	0.08	0.08	0.02	0.06	0.03	0.03	0.02	0.03
MEMIT	0.99	0.97	0.98	0.98	0.68	0.67	0.73	0.69	0.03	0.03	0.01	0.02	0.00	0.00	0.00	0.00
DEFER	0.82	0.86	0.68	0.79	0.66	0.67	0.45	0.59	0.33	0.32	1.00	0.55	0.31	0.31	1.00	0.54
GRACE	1.00	0.46	1.00	0.82	1.00	0.42	1.00	0.81	1.00	0.39	1.00	0.80	1.00	0.37	1.00	0.79
WISE	0.92	0.84	1.00	0.92	0.78	0.74	0.98	0.83	0.62	0.60	1.00	0.74	0.66	0.64	1.00	0.77
AlphaEdit	0.98	0.89	1.00	0.96	0.93	0.85	0.98	0.92	0.91	0.79	0.94	0.88	0.84	0.77	0.56	0.72
MEMOIR	1.00	1.00	1.00	1.00	0.97	0.89	1.00	0.95	0.96	0.89	1.00	0.95	0.94	0.85	1.00	0.93
RECON	1.00	1.00	1.00	**1.00**	0.98	0.91	1.00	**0.96**	0.97	0.89	1.00	**0.95**	0.96	0.86	1.00	**0.94**
Mistral-7B																
FT	0.58	0.54	0.91	0.68	0.39	0.39	0.50	0.43	0.11	0.10	0.02	0.08	0.16	0.13	0.01	0.10
ROME	0.79	0.77	0.98	0.85	0.58	0.57	0.75	0.63	0.05	0.05	0.02	0.04	0.04	0.04	0.02	0.03
MEMIT	0.81	0.79	0.99	0.86	0.46	0.45	0.61	0.51	0.00	0.00	0.01	0.00	0.04	0.04	0.02	0.03
DEFER	0.59	0.68	0.79	0.69	0.44	0.43	1.00	0.62	0.40	0.40	1.00	0.60	0.40	0.39	1.00	0.60
GRACE	1.00	0.36	1.00	0.79	1.00	0.15	1.00	0.72	1.00	0.15	1.00	0.72	1.00	0.02	1.00	0.67
WISE	0.98	0.97	1.00	0.98	0.92	0.89	1.00	0.94	0.87	0.80	1.00	0.89	0.70	0.67	1.00	0.79
AlphaEdit	0.83	0.77	1.00	0.87	0.87	0.75	0.99	0.87	0.86	0.74	0.95	0.85	0.85	0.72	0.68	0.75
MEMOIR	1.00	0.99	1.00	1.00	0.97	0.94	1.00	0.97	0.95	0.91	1.00	0.95	0.94	0.89	1.00	0.94
RECON	1.00	0.99	1.00	**1.00**	0.98	0.94	1.00	**0.97**	0.98	0.91	1.00	**0.96**	0.96	0.89	1.00	**0.95**

Table 3: Extreme-scale sequential editing on ZsRE with LLaMA-3-8B, where T is the length of a single continuous editing sequence with no reset between edits.

	$T = 7{,}000$				$T = 15{,}000$			
Method	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
MEMIT	0.00	0.00	0.00	0.00	–	–	–	–
GRACE	1.00	0.31	1.00	0.77	0.99	0.28	1.00	0.76
WISE	0.59	0.57	1.00	0.72	0.43	0.41	1.00	0.61
AlphaEdit	0.02	0.02	0.00	0.01	0.02	0.02	0.00	0.01
MEMOIR	0.92	0.85	0.99	0.92	0.87	0.78	0.98	0.88
RECON	0.95	0.84	1.00	**0.93**	0.94	0.82	1.00	**0.92**

4.4 Correcting Undesirable Model Behavior

Table 4 corrects undesirable model behavior on SelfCheckGPT. The protocol differs from the previous benchmarks: reliability is the perplexity of the desired correction, locality is token exact match, and no generalization metric is defined. This structurally disadvantages RECON, whose largest gains lie in retrieval under rephrasing, so the comparison is conservative. On Mistral-7B, RECON attains the lowest perplexity at every scale (tied at $T = 1$; 1.15 versus 1.22 for MEMOIR at $T = 600$) with locality at 1.00. On LLaMA-3-8B, it ties MEMOIR at $T = 1$, trails by one hundredth at $T = 10$ and $T = 100$ (1.02 versus 1.01, 1.08 versus 1.07), and leads at the longest scale (1.14 versus 1.25), with locality at 1.00 throughout. The readout design thus extends beyond factual question answering.

4.5 Ablation Study

Table 5 removes or replaces one readout component at a time, at the longest grouped scale of the two benchmarks with a generalization metric. Reliability is invariant to the routing variants (edit queries retrieve their own memory under any routing scheme); norm calibration instead raises reliability by 0.8 points, so the remaining differences concentrate in generalization and locality. Replacing semantic routing with hash-overlap retrieval costs 1.2 generalization points on ZsRE paraphrases and collapses temporally shifted generalization from 0.49 to 0.43, reproducing the hash-routed baselines' failure mode (Table 1). Removing the gate raises generalization ($+0.3$ in distribution, and

Table 4: Hallucination correction on SelfCheckGPT with LLaMA-3-8B (left) and Mistral-7B (right). Reliability is the perplexity of the corrected statement, so lower is better, and locality is token-level exact match on unrelated generations; the benchmark defines no paraphrase-generalization metric.

	LLaMA-3-8B								Mistral-7B							
	$T=1$		$T=10$		$T=100$		$T=600$		$T=1$		$T=10$		$T=100$		$T=600$	
Method	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.	Rel.	Loc.
FT	8.24	0.85	8.05e1	0.37	5.72e2	0.13	2.36e5	0.05	2.50e1	0.38	1.00e2	0.03	1.59e3	0.00	–	–
ROME	1.52	0.97	2.02e1	0.61	1.26e5	0.02	4.07e4	0.02	2.04	0.99	3.45	0.92	1.04e2	0.03	2.41e2	0.01
MEMIT	1.30	0.99	4.05e2	0.93	4.56e10	0.01	2.95e6	0.00	1.64	1.00	1.59e1	0.89	9.72e1	0.04	1.32e2	0.02
DEFER	5.69e2	0.83	2.36e2	0.82	7.02e1	1.00	6.85e1	1.00	1.46e1	0.94	3.22e1	0.99	3.22e1	0.99	3.22e1	1.00
GRACE	1.05	1.00	7.10e1	1.00	7.12e1	1.00	7.73e1	1.00	1.39	1.00	5.97	1.00	9.53	1.00	9.57	1.00
WISE	4.93e1	0.98	1.46	0.95	2.10	0.99	3.20	0.99	1.40	1.00	2.56	0.94	1.31	0.99	5.21	0.93
AlphaEdit	1.58	1.00	3.12	0.98	5.97	0.93	8.49e3	0.05	1.75	1.00	1.76	1.00	2.87	0.98	1.70e2	0.88
MEMOIR	1.00	1.00	1.01	1.00	1.07	1.00	1.25	1.00	1.00	1.00	1.02	1.00	1.09	1.00	1.22	1.00
RECON	**1.00**	**1.00**	1.02	**1.00**	1.08	**1.00**	**1.14**	**1.00**	**1.00**	**1.00**	**1.01**	**1.00**	**1.05**	**1.00**	**1.15**	**1.00**

Table 5: Component ablation at the longest grouped scale of the two benchmarks with a generalization metric: each row removes or replaces one readout component, holding the rest fixed.

	ZsRE, LLaMA-3-8B $T=1{,}000$			Temporal, GPT-J-6B $T=75$		
Configuration	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.
RECON (full)	0.960	0.858	1.000	0.964	0.489	1.000
w/o norm calibration	0.952	0.850	1.000	0.963	0.488	1.000
w/o semantic routing	0.960	0.846	1.000	0.964	0.430	1.000
w/o gate	0.960	0.861	0.713	0.964	0.803	0.701

to 0.80 under shift) but collapses locality (0.713 on ZsRE, 0.70 on Temporal): without an acceptance test, unrelated queries spuriously execute stored memories. Norm calibration costs 0.6 average points when removed at T=1,000 (paired runs report $+0.55$, Table 11), is inactive at $T = 75$ where collisions are rare, and grows monotonically with collision density, reaching 0.7 and 0.8 points at $T = 7{,}000$ and $T = 15{,}000$.

Three observations tie the mechanisms to their failure modes. With the gate disabled, paraphrase generalization remains ≈ 0.99 at $T = 1$, essentially the single-edit ceiling: residuals transfer across reformulations, while ungated locality collapses (0.73 at $T = 1{,}000$, 0.53 at $T = 15{,}000$). The similarity structure is strictly separated (Figure 1a): the highest-scoring unrelated query scores 0.38, versus 0.42 for the hardest paraphrase. The selected $\tau = 0.55$ sits on the conservative edge of the plateau of Table 7, trading 0.3 generalization points for a wider safety margin. Calibration is collision-specific: Table 8 shows a single-peaked sweep at $T = 1{,}000$, and Figure 3 a flat response at $T = 20$, where collisions are rare.

5 Conclusion

We studied lifelong editing in the residual-memory paradigm and traced degradation at scale to the readout rather than the write. RECON repairs this readout without training: semantic routing–support decoupling retrieves memories through a gated dense semantic key while executing each sparse support verbatim, and retrieval-conditioned norm calibration rescales retrieved dimensions in closed form. Across Temporal, ZsRE, and SelfCheckGPT, RECON nearly doubles out-of-distribution generalization over the hash-based readout at T=10 and preserves locality through $15{,}000$ edits, suggesting scalability is governed less by how updates are stored than by how they are retrieved and executed.

AI USE STATEMENT

We used generative AI tools to assist with the statistical analysis of the data presented in this work. The AI-assisted analyses were reviewed and verified by the authors against the original data and experimental results.

We also used generative AI tools to improve the grammar, clarity, and readability of the manuscript. All AI-assisted text was reviewed and revised by the authors.

The authors are responsible for the final content of this work, including all analyses, claims, and written content.

REPRODUCIBILITY STATEMENT

We have taken the following steps to support reproducibility. The RECON method, including semantic routing with support-preserving execution and retrieval-conditioned norm calibration over the residual memory store, together with the training-free inference procedure, is described in Section 3. Datasets and the editing protocol, evaluation metrics, baseline methods, backbone models, and implementation details are specified in Section 4.1 and Appendix A. The full hyperparameter configurations, including the semantic gate threshold τ and the calibration strength α at each editing scale, are listed in Appendix B. Statistical significance tests, retrieval diagnostics, and latency measurements are reported in Appendix C. We will include anonymous source code and configuration files with the supplementary material to reproduce the editing pipeline, evaluation, and all reported results.

REFERENCES

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pp. 6491–6506. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.522.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 13565–13580. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.751.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*. OpenReview.net, 2025.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36, NeurIPS 2023*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas

Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning, CoNLL 2017*, pp. 333–342. Association for Computational Linguistics, 2017. doi: 10.18653/V1/K17-1034.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *Forty-second International Conference on Machine Learning, ICML 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR / OpenReview.net, 2024.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 9004–9017. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.557.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022a.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022b.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 billion parameter autoregressive language model trained on The Pile. *arXiv preprint arXiv:2104.09864 / GitHub: kingoflolz/mesh-transformer-jax*, 2021.

Ke Wang, Yiming Qin, Nikolaos Dimitriadis, Alessandro Favero, and Pascal Frossard. MEMOIR: lifelong model editing with minimal overwrite and informed retention for llms. In *Advances in Neural Information Processing Systems 38, NeurIPS 2025*, 2025.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems 37, NeurIPS 2024*, 2024a.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2024*, pp. 82–93. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.ACL-DEMOS.9.

Renzhi Wang and Piji Li. Lemoe: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 2551–2575. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.149.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 10222–10240. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.632.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: enhancing model editing with neuron-indexed dynamic lora. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, pp. 19449–19457. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29916.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286, 2024. doi: 10.48550/ARXIV.2401.01286.

A Experimental Details

A.1 Datasets and Editing Protocol

Datasets. ZsRE (Levy et al., 2017) provides question-form edits derived from Wikidata relations for zero-shot relation extraction; each edit ships with a paraphrase probe and a neighborhood probe. The standard pool contains $1{,}000$ edits; the extreme-scale setting extends it to single sequences of $7{,}000$ and $15{,}000$ edits on LLaMA-3-8B, drawing the additional edits from the dataset's train split. Temporal evaluates OOD temporal editing with GPT-J-6B under the construction of Wang et al. (2025): each edit pairs the opening paragraph of a Wikipedia entity page with a GPT-4-generated passage about that entity, and generalization is probed on the remainder of that entry, so the test distribution differs from the editing distribution. Locality probes are drawn from the Pile (Gao et al., 2021), on which GPT-J-6B was pre-trained. The pool contains 75 edits. SelfCheckGPT (Manakul et al., 2023) targets hallucination correction: passages about Wikidata entities are sampled from GPT-3 and annotated for factual hallucination, and each edit pairs a passage with a correction verified against Wikipedia; the pool contains 600 edits.

Protocol. We follow the grouped evaluation protocol of Wang et al. (2025). The edit pool is partitioned into groups of T sequential edits; the model is reset between groups, and all metrics for a group are measured after the group's full edit sequence has been applied. Where T does not divide the pool (Temporal at $T = 10$), the final group is shortened (eight groups, the last with five edits); every pool case is still evaluated exactly once. Settings with few edits per group average over all groups of the pool, each built from different edits (e.g., $1{,}000$ single-edit groups on ZsRE).

Benchmarks. The three benchmarks are complementary by design: Temporal maximizes the distribution shift between editing and evaluation, ZsRE provides the standard grouped sequential protocol under which all baselines are directly comparable, and SelfCheckGPT already requires overriding beliefs the model currently holds. Counterfactual rewriting against strong parametric priors, as in CounterFact (Meng et al., 2022), is outside the present scope.

A.2 Dataset Examples

We show one real sample per benchmark, with the fields as consumed by the editing pipeline.

ZsRE. Edit prompt: "When was the inception of the IAAF Combined Events Challenge?" desired completion: "2006"; paraphrase probe: "When was the IAAF Combined Events Challenge launched?"; neighborhood probe: "nq question: what is the name of the last episode of spongebob" with expected answer "The String".

Temporal. Edit prompt: "Tesla, Inc.,"; desired completion: "originally known as Tesla Motors, is an American electric vehicle and clean energy company founded by Martin Eberhard, Marc Tarpenning, Ian Wright, JB Straubel, and most notably, Elon Musk. [. . .]"; temporally shifted probe: "Tesla, Inc., an American multinational automotive and clean energy company headquartered in Austin, Texas, [. . .]".

SelfCheckGPT. Edit prompt: "This is a Wikipedia passage about carole gist. Carole Gist (born April 28, 1969) is an American beauty pageant titleholder from Detroit, Michigan who was crowned Miss USA 1990. [. . .]"; desired correction: "She was also the first contestant from Michigan to win Miss USA, and broke the five-year streak of winners from Texas."; locality probe: an unrelated passage ("Description Map of South America. [. . .]") with its ground-truth continuation, scored by token exact match.

A.3 Evaluation Metrics

We adopt the standard three-metric protocol of lifelong editing (Wang et al., 2024a), implemented with EasyEdit (Wang et al., 2024b): reliability measures success on the edit prompt, generalization measures transfer to rephrased or shifted probes, and locality measures preservation on irrelevant samples.

ZsRE. Answers are greedily generated and scored by exact match after normalization (case folding; removal of articles and punctuation). Reliability scores the edit prompt; generalization scores the paraphrase probe; locality scores the neighborhood probe. Per-case scores are averaged over probes and then over cases, and the average is the mean of the three metrics.

Temporal. Following the probability-threshold criterion of Wang et al. (2025), a desired-completion token counts as correct when its per-token loss is below $-\log 0.8$, so success does not hinge on regenerating one exact continuation. Reliability aggregates the criterion over the edit prompt's completion; generalization aggregates it over the shifted probe's completion; locality is token exact match on neighborhood continuations; the average is the mean of the three.

SelfCheckGPT. Reliability is the teacher-forced perplexity of the desired correction given the passage, computed on the correction tokens only (lower is better), and locality is token exact match on the continuation of an unrelated passage, following Wang et al. (2024a). No generalization metric exists in this setting and no average is reported, as perplexity and accuracy are not commensurable (Wang et al., 2025).

A.4 Baseline Methods

All baselines are evaluated under the grouped protocol of Section 4.1; their numbers are transferred from Wang et al. (2025), who reproduce the baselines with EasyEdit (Wang et al., 2024b) under the identical protocol, so no re-implementation gap arises. The baselines span three families.

In-place parameter editors. FT fine-tunes the model parameters on each edit. ROME (Meng et al., 2022) locates the MLP projection that stores a fact via causal tracing and writes a closed-form rank-one update to it. MEMIT (Meng et al., 2023) extends this to batches of edits by distributing closed-form updates across layers. AlphaEdit (Fang et al., 2025) constrains each update to the null space of preserved knowledge.

Retrieval-gated memories. GRACE (Hartvigsen et al., 2023) maintains a discrete key–value codebook over intermediate activations and overwrites the activation when a key matches, resolving conflicts by splitting keys. WISE (Wang et al., 2024a) attaches a trained side memory and routes

each query between the base model and the side memory with a learned router. DEFER denotes the SERAC-based editor of Mitchell et al. (2022b), adapted to the lifelong setting by Hartvigsen et al. (2023). GRACE and WISE are themselves dynamic-routing designs, so the comparison spans retrieval-gated memory routing rather than only hash-based lookup.

Sparse residual storage. MEMOIR (Wang et al., 2025) is the strongest residual-memory editor published so far: it stores each edit as a sparse residual over a hash-allocated support and retrieves by hash overlap through an acceptance gate. It is the hash-based readout baseline whose failure modes RECON repairs.

Where a baseline is evaluated under a protocol directly comparable to ours, its numbers in Table 2 are transcribed from Wang et al. (2025).

A.5 Backbone Models

We use three decoder-only backbones, following Wang et al. (2025). LLaMA-3-8B-Instruct (Team, 2024) and Mistral-7B (v0.1) (Jiang et al., 2023) (32 layers, hidden size $4{,}096$) serve the ZsRE and SelfCheckGPT benchmarks; GPT-J-6B (Wang & Komatsuzaki, 2021) (28 layers, hidden size $4{,}096$) serves Temporal. We refer to LLaMA-3-8B-Instruct as LLaMA-3-8B in the main text.

Residual-memory editing targets a single projection by design: all edits accumulate into one shared residual matrix ΔW (Section 3.1), which is what keeps parameter storage independent of the editing horizon. The remaining architectural choice is the layer, which we select by sweeping the adjacent down-projections on ZsRE with LLaMA-3-8B (layers 23, 25, 27, and 29): layer 27 attains the best paraphrase generalization, and a check against its closest rival at T=1,000 confirms the choice. We keep it for Mistral-7B, which shares the architecture, and edit GPT-J-6B at layer 21 (`fc_out`). For LLaMA-3-8B and Mistral-7B the target projection maps $\mathbb{R}^{14{,}336} \to \mathbb{R}^{4{,}096}$, so $D = 14{,}336$ and the sparse supports cover $\kappa = 1{,}536$ columns ($\approx 11\%$ of D) at $T \geq 10$, with $\kappa = 4{,}096$ inherited from the base editor at $T = 1$ (Table 6). For GPT-J-6B the target projection maps $\mathbb{R}^{16{,}384} \to \mathbb{R}^{4{,}096}$, so $D = 16{,}384$ and $\kappa = 4{,}096$ (25% of D).

A.6 Implementation Details

All readout components are computed from the frozen backbone and involve no training. The background mean μ averages the prompt-aggregated hidden states of 100 edit-irrelevant prompts (the unrelated sample interleaved with each editing batch), precomputed once per edited layer and held fixed across all experiments. The calibration applies a numerical floor $\epsilon = 10^{-6}$ to the column norms. Edits are applied one at a time (batch size one), following the lifelong editing setting (Hartvigsen et al., 2023; Wang et al., 2024a). The write phase optimizes the supported columns of ΔW with SGD (weight decay 10^{-5}; learning rates in Table 6) to minimize the cross-entropy of the target completion y_i on the edit prompt e_i, exactly as in the base editor.

B Hyperparameter Analysis

RECON introduces two hyperparameters: the semantic gate threshold τ and the calibration strength α; Table 6 lists the per-scale configurations used for the main results. Because both mechanisms target degradation that emerges as stored edits collide, we analyze them at $T = 1{,}000$ on ZsRE with LLaMA-3-8B, the longest horizon of the standard sequential-editing protocol and the scale where collision effects are strongest, varying one parameter at a time around the selected configuration ($\tau = 0.55$, $\alpha = 0.125$). Tables 7 and 8 report the two sweeps, and Figure 3 contrasts the calibration sweep with $T = 20$, where collisions are rare.

Table 6: Per-scale RECON configurations used for the main results. The write-side settings (edit learning rate, iteration budget n_{iter}, support size κ, and the target layer), together with the base editor's hash-overlap gate threshold, are inherited unchanged; τ and α are the two hyperparameters RECON introduces. The T=1,000 ZsRE row also covers the single-sequence runs at T=7,000 and T=15,000.

Benchmark	Model	T	τ	α	lr	n_{iter}	κ	irr	Layer
ZsRE (edit pool 1,000, grouped protocol)									
	LLaMA-3-8B	1	0.35	0.025	1.0	30	4,096	0.40	27 `down_proj`
	LLaMA-3-8B	10	0.45	0.125	1.0	30	1,536	0.40	27 `down_proj`
	LLaMA-3-8B	100	0.45	0.125	2.0	30	1,536	0.40	27 `down_proj`
	LLaMA-3-8B	1000	0.55	0.125	1.0	30	1,536	0.40	27 `down_proj`
	Mistral-7B	1	0.40	0.025	1.0	70	4,096	0.40	27 `down_proj`
	Mistral-7B	10	0.35	0.125	1.0	70	1,536	0.40	27 `down_proj`
	Mistral-7B	100	0.45	0.125	1.0	70	1,536	0.40	27 `down_proj`
	Mistral-7B	1000	0.50	0.125	1.0	70	1,536	0.40	27 `down_proj`
SelfCheckGPT (edit pool 600, grouped protocol; τ fixed)									
	LLaMA-3-8B	1	0.55	0.125	1.0	30	1,536	0.40	27 `down_proj`
	LLaMA-3-8B	10	0.55	0.125	1.25	30	1,536	0.40	27 `down_proj`
	LLaMA-3-8B	100	0.55	0.125	1.0	30	1,536	0.40	27 `down_proj`
	LLaMA-3-8B	600	0.55	0.125	1.0	30	1,536	0.40	27 `down_proj`
	Mistral-7B	1	0.55	0.125	1.0	70	1,536	0.40	27 `down_proj`
	Mistral-7B	10	0.55	0.125	1.0	70	1,536	0.40	27 `down_proj`
	Mistral-7B	100	0.55	0.125	1.0	70	1,536	0.40	27 `down_proj`
	Mistral-7B	600	0.55	0.25	1.0	70	1,536	0.40	27 `down_proj`
Temporal (edit pool 75, grouped protocol)									
	GPT-J-6B	10	0.35	0.125	1.0	70	4,096	0.46	21 `fc_out`
	GPT-J-6B	75	0.45	0.125	1.0	70	4,096	0.46	21 `fc_out`

Table 7: Sweep of the semantic gate threshold τ on ZsRE at T=1,000 with LLaMA-3-8B, varying τ with the other hyperparameter held fixed.

τ	Rel.	Gen.	Loc.	Avg.
0.10	0.960	0.861	0.787	0.870
0.20	0.960	0.861	0.951	0.924
0.30	0.960	0.861	0.996	0.939
0.35	0.960	0.861	0.999	0.940
0.40	0.960	0.861	1.000	0.940
0.45	0.960	0.860	1.000	0.940
0.50	0.960	0.859	1.000	0.940
0.55	0.960	0.858	1.000	0.940
0.60	0.960	0.845	1.000	0.935
0.65	0.960	0.827	1.000	0.929
0.70	0.960	0.785	1.000	0.915

Table 8: Sweep of the calibration strength α on ZsRE at T=1,000 with LLaMA-3-8B, varying α with the other hyperparameter held fixed.

α	Rel.	Gen.	Loc.	Avg.
0	0.952	0.850	1.000	0.934
0.025	0.955	0.853	1.000	0.936
0.05	0.957	0.855	1.000	0.937
0.1	0.960	0.857	1.000	0.939
0.125	0.960	0.858	1.000	0.940
0.25	0.962	0.849	1.000	0.937
0.5	0.924	0.785	1.000	0.903

Hyperparameter schedule. The retrieval-side settings follow a single variable: the collision density of the residual store. On ZsRE and Temporal, the gate threshold is low at short horizons ($\tau \in [0.35, 0.45]$ for $T \leq 100$), where paraphrase recall is the bottleneck—paraphrased queries score below exact prompts while neighborhood probes stay below the lowered threshold, so the gate buys generalization at no locality cost—and rises at long horizons ($\tau \in [0.50, 0.55]$ at $T = 1{,}000$), where crosstalk among co-stored edits becomes the dominant error source; the Mistral $T = 1{,}000$ cell stops at 0.50 because paraphrase recall is still the bottleneck there, and the gentler threshold gains 0.55 generalization points at no reliability cost. SelfCheckGPT keeps $\tau = 0.55$ throughout: the evaluation re-presents the edited passage itself, so every reliability probe matches its own memory and the gate passes at any threshold. The calibration exponent follows the same axis: $\alpha = 0.025$ at $T = 1$, where columns are rarely overwritten, and 0.125 once collisions accumulate

(the $T = 1{,}000$ dose–response peak; lowering it at $T = 100$ costs 0.16 average points), rising to 0.25 only in the densest collision cell (Mistral-7B at SelfCheckGPT $T = 600$). On LLaMA-3-8B and Mistral-7B, the residual support is wider only in the single-write regime ($\kappa = 4{,}096$ at $T = 1$ versus 1,536 at $T \geq 10$): under collisions, a wider support dilutes exact-prompt retrieval and lowers the average at $T = 10$, 100, and 1,000 alike. The write loop and its learning rate are inherited unchanged except in two cells (LLaMA-3-8B at ZsRE $T = 100$ and SelfCheckGPT $T = 10$, raised to 2.0 and 1.25), where the bottleneck is transfer strength rather than crosstalk.

Gate threshold. Table 7 sweeps the gate threshold across $\tau \in [0.10, 0.70]$, and reliability is exactly invariant: edit queries retrieve their own memory under any threshold, so τ never affects what is written or whether it is found. The remaining two metrics expose opposite edges of the operating range. Paraphrase generalization declines monotonically with τ, but negligibly over the permissive range (from 0.861 at $\tau \leq 0.40$ to 0.858 at $\tau = 0.55$) before stricter gates cut paraphrase recall sharply ($0.861 \rightarrow 0.785$, a 7.6-point drop from the plateau to $\tau = 0.70$). Locality is untouched down to $\tau = 0.40$ and then collapses progressively as the gate opens (0.996 at $\tau = 0.30$, 0.951 at 0.20, 0.787 at 0.10), approaching the gate-free limit of 0.73 measured on the ungated readout; generalization is still on its plateau at $\tau = 0.10$, so the threshold trades locality against paraphrase recall on two disjoint sides. All three metrics simultaneously sit on their plateau across $\tau \in [0.40, 0.55]$, and the selected value lies inside this window on its conservative side. On LLaMA-3-8B the ZsRE schedule raises τ with the store size (from 0.35 at $T = 1$ to 0.55 at $T = 1{,}000$, Table 6): as more keys accumulate, the best-match similarity attainable by unrelated queries rises, so a threshold well above the rising floor keeps locality at 1.00 with margin. The plateau itself persists at extreme scale: at $T = 7{,}000$, $\tau \in \{0.45, 0.50, 0.55\}$ yields averages of 0.931, 0.931, and 0.930 with locality at or above 0.999 (Table 9).

Table 9: Gate-threshold sweep at extreme scale: ZsRE with LLaMA-3-8B in a single continuous sequence of $T = 7{,}000$ edits.

τ	Rel.	Gen.	Loc.	Avg.
0.45	0.953	0.840	0.999	0.931
0.50	0.953	0.839	1.000	0.931
0.55	0.953	0.836	1.000	0.930

Calibration strength. Table 8 sweeps the calibration exponent, which follows an inverted-U response: relative to the uncalibrated readout ($\alpha = 0$), every value in $\alpha \in [0.025, 0.25]$ improves performance, rising monotonically to the selected $\alpha = 0.125$ ($+0.55$ average points) before degrading toward full column normalization ($\alpha = 1$); at the intermediate $\alpha = 0.5$ the average already falls 3.1 points below the uncalibrated readout. The selected value is thus not a fragile point estimate but the interior peak of a smooth, single-peaked response.

Table 11 traces the calibration gain back to collision density: it is worth exactly zero points at $T = 20$, where residual columns receive about two writes, and grows to $+0.55$ points at $T = 1{,}000$, where they receive about one hundred; the $T = 200$ cell is a single-group estimate within noise of the trend. The calibration is thus specific to collision-induced distortion rather than acting as generic regularization. In the densest collision cell we evaluate—SelfCheckGPT on Mistral-7B at $T = 600$,

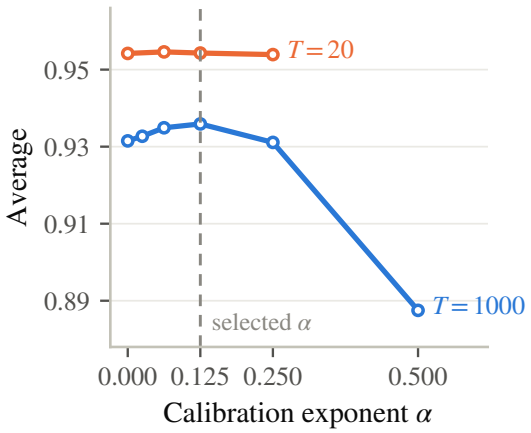

Figure 3: Calibration sweep of the average score at $T = 1{,}000$ and $T = 20$ on ZsRE with LLaMA-3-8B. Absolute levels differ from Table 8 by up to 0.4 points across runs; the response shape is invariant.

roughly 64 writes per edited column—the response stays monotone (Table 10): perplexity falls from 1.19 at $\alpha = 0.0625$ to 1.15 at the selected $\alpha = 0.25$.

Table 10: Calibration dose–response in the densest collision cell: SelfCheckGPT on Mistral-7B at $T = 600$ (about 64 writes per edited column). Reliability is the perplexity of the corrected statement, so lower is better.

α	Rel.	Loc.
0.0625	1.188	0.998
0.125	1.178	0.998
0.1875	1.166	0.998
0.25	1.154	0.998

Table 11: Paired effect of norm calibration across editing scales on ZsRE with LLaMA-3-8B: average score of the calibrated readout against the uncalibrated one. The parenthesized value approximates the writes per residual column; $T = 20$ uses an edit pool of 300, $T = 200$ a pool of 200, and the remaining scales the full pool of 1,000. Rows come from dedicated paired runs whose absolute levels differ from the main tables by up to 0.7 points; within-table comparisons are exact.

T (writes/column)	Uncalibrated	RECON	Δ
20 (≈2)	0.9609	0.9609	0.00
100 (≈11)	0.9451	0.9468	+0.17
200 (≈21)	0.9501	0.9508	+0.07
1,000 (≈107)	0.9341	0.9396	+0.55

C Statistical Significance and Retrieval Diagnostics

Paired significance tests. At $T = 1{,}000$ on ZsRE with LLaMA-3-8B, we pair all 1,000 evaluation cases by index across arms and apply two-sided Wilcoxon signed-rank tests. Against the strongest residual-memory baseline's readout, RECON's reliability margin is $+2.5$ points (78 wins, 19 losses, 903 ties; $p = 3.1 \times 10^{-10}$) and its average margin is $+1.1$ points ($p = 1.5 \times 10^{-4}$). The effect of norm calibration against the uncalibrated readout ($\alpha = 0$) is significant on every metric: reliability $+0.8$ points ($p = 2.4 \times 10^{-5}$), generalization $+0.8$ points ($p = 2.4 \times 10^{-3}$), and average $+0.5$ points on the paired per-case mean (the difference of the table-level averages is $+0.55$, Table 11; $p = 2.8 \times 10^{-6}$), with locality identical in all 1,000 pairs.

Retrieval diagnostics. Query–key similarities can be recomputed exactly from the frozen backbone, because query features read the input side of the edited projection and are therefore invariant to the edit state. At the full $T = 1{,}000$ store, target prompts match their own key with cosine exactly 1.00 and are ranked first in every case: the reliability ceiling is set entirely by residual execution, not retrieval. Unrelated prompts have a median best-match similarity of 0.17 and none of them exceeds the gate threshold, so locality is structurally protected at this scale.

Oracle routing. Table 12 decomposes the $T = 1{,}000$ generalization gap. Forcing each paraphrase query to its source memory—oracle routing applied only to paraphrase queries, with the gate forced open for those queries—raises generalization from 0.858 to 0.922, while reliability and locality, evaluated under the standard gated readout, stay identical at 0.960 and 1.00: oracle forcing touches only queries that retrieval would otherwise route wrongly, confirming the routing invariance of reliability at the full store. The gap decomposes into two parts. Relaxing the threshold alone ($\tau = 0.30$) recovers only 0.3 points, so gate rejection is marginal; the remaining ≈ 6 points are queries whose keys rank another memory first. The margin between the oracle ceiling (0.922) and single-edit generalization (0.99) lies on the execution side, where norm calibration operates. Retrieval, gating, and calibration thus each govern a distinct share of the readout gap.

Table 12: Decomposition of the generalization gap at $T = 1{,}000$ on ZsRE with LLaMA-3-8B: the gated readout, the same readout with a relaxed threshold, and oracle routing that forces each paraphrase query to its source memory with the gate open for those queries; reliability and locality probes keep the standard gated readout.

Retrieval configuration	Rel.	Gen.	Loc.
Gated top-1 retrieval (τ=0.55)	0.960	0.858	1.000
Relaxed threshold (τ=0.30)	0.960	0.861	0.996
Oracle routing (gate open)	0.960	0.922	1.000

Scaling of component contributions. Repeating the component ablation at the extreme scales with single continuous sequences confirms the same pattern. Removing norm calibration costs 0.71 and 0.84 average points at $T = 7{,}000$ and $T = 15{,}000$, extending the effect measured at grouped scales and confirming that the calibration gain grows monotonically with collision density. Removing the gate leaves generalization almost unchanged but collapses locality to 0.60 and 0.53—at $15{,}000$ stored memories, about half of all unrelated queries would spuriously execute a stored memory. Replacing semantic routing with hash-overlap retrieval is nearly neutral at these in-distribution scales: its benefit concentrates on out-of-distribution transfer, as the Temporal panel of Table 5 shows, while lexical overlap keeps hash retrieval competitive in distribution (Section 4.3). At the shorter Temporal scale the effect is far larger: at $T = 10$, the same replacement drives temporally shifted generalization from 0.72 to 0.40 at unchanged reliability (0.96; Table 13)—consistent with the hash-routed baseline's 0.37 in Table 1—so the out-of-distribution benefit of semantic routing holds at every Temporal scale.

Table 13: Out-of-distribution component ablation on Temporal with GPT-J-6B: replacing semantic routing with hash-overlap retrieval collapses temporally shifted generalization at unchanged reliability, at both grouped scales.

T	Configuration	Rel.	Gen.	Loc.
10	RECON	0.959	0.721	1.000
	w/o semantic routing	0.959	0.399	1.000
75	RECON	0.964	0.489	1.000
	w/o semantic routing	0.964	0.430	1.000

Latency. The exhaustive routing search is a single $T \times D$ matrix–vector product. Table 14 reports per-query latency measured end to end on a single RTX 4090, with synthetic memories injected so that the readout path is exercised at full store sizes: routing adds negligible overhead up to T=1,000 and grows linearly in T, norm calibration adds no measurable cost, and the search is recoverable with approximate nearest-neighbor indexes over the static keys.

Table 14: Per-query inference latency (ms) on a single RTX 4090, measured end to end with synthetic memories injected at full store sizes.

	$T = 1{,}000$	$T = 15{,}000$
Semantic routing (ours)	0.73	12.8
Hash-overlap scoring	–	2.24
One backbone forward pass	29.6	29.6
Total per query	30.3	42.4